UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/759,737	01/16/2004	Jeffrey P. Snover	MS1-1902US	6905

22801 7590 02/13/2008

LEE & HAYES PLLC
421 W RIVERSIDE AVENUE SUITE 500
SPOKANE, WA 99201

EXAMINER	
ZHEN, LI B	
ART UNIT	PAPER NUMBER
2194	
MAIL DATE	DELIVERY MODE
02/13/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/759,737	SNOVER ET AL.	
	Examiner	Art Unit	
	Li B. Zhen	2194	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *19 November 2007*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-27 and 29-32* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-27 and 29-32* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *07/18/2007,01/18/2008*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1 – 27 and 29 – 32 are pending in the application.

Response to Arguments

2. Applicant's arguments filed 11/19/2007 have been fully considered but they are not persuasive. In response to the Non-Final office action dated 05/18/2007, applicant argues:

(1) Russell itself says that it requires a default constructor because Russell discloses that a default constructor "must" be used. The default constructor is the "pre-defined definition of the specified data type of the object." [pp. 11 and 14];

(2) As claimed, the "decomposing" and "serializing" acts without reliance upon the "pre-defined definition of the specified data type of the object." [p. 14];

(3) Russell creates its customized serializer in response to a failure of a target object (i.e., instance of an object) to match its pre-defined definition for its object type. For example, Russell uses traditional reflection (which is described briefly in App. p. 18, lines 17-18). Russell identifies the data type of target object is identified as X. If, for example, the pre-defined definition of data type X has three properties, but the actual target has four properties, Russell will fail. [pp. 14 – 15];

(4) For Russell to function properly there must be a default constructor that matches the properties in the target object. [p. 15];

(5) At this time, Applicant makes no comment about whether the Examiner's premise (i.e., "serialization process requires calling reflection on an object") or

conclusion (i.e., "objects in applicant's invention would also require a default constructor") is true. It seems that the Examiner should provide proof that its premise is true before the Applicant can respond to the conclusion. [p. 15];

(6) Even if the premise and conclusion is true, Applicant submits the following: (a) By it own admission (paragraph [0072], Russell requires a default constructor because Russell discloses that a default constructor "must" be used; (b) The default constructor is a "pre-defined definition of the specified data type of the object"; (c) One or more claims recite "decomposing" and "serializing" that acting without reliance upon "pre-defined definition of the specified data type of the object," thus without reliance on a "default constructor." [pp. 15 – 16]

As to argument (1), examiner disagrees and notes that a "default constructor" is not a "pre-defined definition of a specified data type of an object". According to "Data Structures and Algorithms with Object-Oriented Design Patterns in C++" [hereinafter Preiss, discussed in applicant's response dated 11/19/2007; p. 14, paragraph (009)], a default constructor is a constructor that takes no arguments. A constructor is a member function that has the same name as its class that initializes an object [page598.html of Preiss]. Therefore a default is a class member function with the same name as its class that takes no arguments and initializes an object. The default constructor initializes the member variables of the class [p. 599, lines 5 – 7] and does not define the data type of the object. The resources identified on p. 14 of applicant response do not define a "default constructor" as a "pre-defined definition of a specified data type of an object".

According to Preiss, the class construct is used to define specific data types [C++ supports the creation of user-defined types using the class construct, p. 596]. Applicant's specification does not provide a definition for a "default constructor". In fact, examiner was unable to locate any reference to any type of constructor in applicant's specification. Therefore, the terms in the claims will be given its ordinary meaning known to one reasonably skilled in the art.

As to argument (2), examiner notes that the claims are given it broadest reasonable interpretation and notes that the "pre-defined definition of the specified data type of the object" is not interpreted as a default constructor [see response to argument (1) above]. The default constructor is a function that initializes an object during instantiation and not a "pre-defined definition of the specified data type of the object". The serialization process in Russell requires a function that initializes an object during object instantiation [default constructor] and does not require a "pre-defined definition of the specified data type of the object". The section that applicant that points to in the Russell reference [paragraph 0072] discloses invoking the default constructor of embedded objects to initialize the objects. Russell discloses the default constructor as a function that is invoked to initialize the object and does not define the default constructor as a "pre-defined definition of the specified data type of the object". Therefore, Russell teaches applicant's invention as claimed.

As to argument (3), examiner was unable to locate the portion of Russell that discloses creating a customized serializer in response to a failure of a target object to match its pre-defined definition for its object type. For example, examiner was unable to locate any disclosure of matching the number of properties of the object with the pre-defined definition of the data type. Russell does not make any reference to object properties or pre-defined definition of a data type.

As to argument (4), examiner was unable to location the portion of Russell that specifically requires a default constructor that matches the properties in the target object. The section that applicant that points to in the Russell reference [paragraph 0072] discloses invoking the default constructor of embedded objects to initialize the objects. Paragraph 0072 of Russell does not disclose the requirement for a default constructor that matches the properties in the target object.

As to argument (5), a constructor is a member function that initializes an object [p. 598 of Preiss]. When an object is instantiated, the constructor or default constructed is invoked to initialize the member variables of the object. In order for an object to exist so that it can be serialized, the object has to be instantiated first (which requires some type of constructor). Based on the applicant's claims and arguments, applicant is either suggesting: (1) the objects in their claims are instantiated without invoking any type of constructor; or (2) the objects in their claims are instantiated by invoking any constructor except for the default constructor. If applicant is suggesting point (1), it is unclear as to

how any object can be instantiated without invoking a constructor of any type. If applicant is suggest point (2), examiner notes that applicant's specification does not specifically disclose invoking a specific type of constructor during object instantiation. In fact, examiner was unable to locate any disclosure regarding constructors of any type.

As to argument (6), examiner agrees with submission (a) and disagrees with submissions (b) and (c). As to submission (b) the default constructor is a "pre-defined definition of the specified data type of the object", see response to argument (1) above. As to submission (c) one or more claims recite "decomposing" and "serializing" that acting without reliance upon "pre-defined definition of the specified data type of the object," thus without reliance on a "default constructor", see response to argument (2) above.

Claim Rejections - 35 USC § 102

3. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

4. **Claims 1 – 8, 11, 13, 14, 16, 17, 26, 27 and 29 – 32 are rejected under 35 U.S.C. 102(e) as being anticipated by U.S. Patent Application Publication No. 2004/0039964 to Russell et al. [hereinafter Russell, previously cited].**

5. As to claim 1, Russell teaches at least one computer-readable storage medium [p. 8, paragraph 0096] having computer executable instructions that provide a method for transferring computer-readable objects [p. 5, paragraph 0071] across a remote boundary [p. 7, paragraph 0088], the method comprising:

without relying on a pre-defined definition of a specified data type of an object [the serializer 220 of preferred embodiments not only does not require this input, but it generates the type mapping 210 "on the fly", as needed to carry out a serialization; p. 3, paragraph 0050], decomposing the object [complex object 500 that needs to be serialized; p. 5, paragraph 0063] into multiple sub-components [A child element 930 is generated for the top-level class of the object being serialized; p. 6, paragraph 0080] wherein the decomposing comprising extracting discernable properties [attributes; p. 6, paragraph 0080] and values for each sub-components [values of these attributes are also generated by parsing the exception and locating the class name; p. 6, paragraph 0075];

without relying on the pre-defined definition of the specified data type of the object [the serializer 220 of preferred embodiments not only does not require this input, but it generates the type mapping 210 "on the fly", as needed to carry out a serialization; p. 3, paragraph 0050], serializing the multiple sub-components and their

discernable properties and values into a serialized package [entire complex object has been marshalled, the result is a serialized object as shown at 540; p. 4, paragraph 0057]; and

transmitting the serialized package to a remote entity [Message 1015 is then sent through the network 1020, 1025, and then reaches the target Web service 1030; p. 7, paragraphs 0087 and 0088 and p. 3, paragraph 0043],

wherein the decomposing, serializing, and transmitting facilities transferring computer-readable objects across a remote boundary [a JavaBean is serialized for transmission to a Web service or other request message consumer; p. 4, paragraph 0059].

6. As to claim 26, this is a system claims that corresponds to product claim 1; note the rejection to claim 1 above, which also meets the limitations of this system claim. Russell also teaches the system comprising a processor [p. 8, paragraph 0097] and memory [p. 8, paragraph 0096].

7. As to claim 29, Russell teaches the decomposing act further comprises: dividing the multiple sub-components into a hierarchy based upon a list of known object types [A child element 930 is generated for the top-level class of the object being serialized; p. 6, paragraph 0080], the known object types being a type known by the remote entity [type mappings; p. paragraph 0077]; and associating an object with one of the known object types, so that object is a known object [serializer next needs to serialize an instance of AccountTransactions class, the type mappings will be available; p. 6, paragraph 0077].

8. As to claim 2, Russell teaches the list identifies the first type as one of the known object types [using selected ones of the stored type mappings when the serialization process is subsequently invoked for the determined object; p. 2, paragraph 0018 and p. 6, paragraph 0077].

9. As to claim 3, Russell teaches at least one sub-component comprises an unknown object having a type unidentified within the list [a "missing" type mapping that causes an exception during serialization is programmatically resolved; p. 2, paragraph 0036].

10. As to claim 4, Russell teaches the decomposing an object further comprises decomposing the unknown object into another level of sub-components based on the list [A recursive process is then invoked to serialize an instance of this class. Once the entire complex object has been marshalled, the result is a serialized object as shown at 540; p. 4, paragraph 0057].

11. As to claim 5, Russell teaches a first process [object-based client code; p. 3, paragraph 0048] on a system transmits the serialized package [p. 4, paragraph 0057] and the remote entity comprises another process [Web service 1030; p. 7, paragraph 0088] on the system [client/server networks; p. 3, paragraph 0043].

12. As to claim 6, Russell teaches a first process [object-based client code; p. 3, paragraph 0048] on a system transmits the serialized package [p. 4, paragraph 0057] and the remote entity comprises another process [Web service 1030; p. 7, paragraph 0088] on another system [client/server networks; p. 3, paragraph 0043].

13. As to claim 7, Russell teaches a first application [object-based client code; p. 3, paragraph 0048] domain [client/server networks; p. 3, paragraph 0043] executing within a process transmits the serialized package [p. 4, paragraph 0057] and the remote entity comprises another application domain [client/server networks; p. 3, paragraph 0043] within the process [Web service 1030; p. 7, paragraph 0088].

14. As to claim 8, Russell teaches the hierarchy comprises a property bag ["mappings" element 800 contains a child "map" element for each mapping being generated for this JavaBean; p. 6, paragraph 0074].

15. As to claim 11, Russell teaches the property bag comprises a plurality of entries ["mappings" element 800 contains a child "map" element for each mapping; p. 6, paragraph 0074], each entry being associated with one of the sub-components [class name in the example is "AccountTransactions"; p. 6, paragraph 0075] and having a first field for storing a name ["qname" attribute; p. 6, paragraph 0075] associated with the sub-component [name of the class; p. 6, paragraph 0075], a second field for storing a value associated with the sub-component [values of these attributes are also generated

by parsing the exception and locating the class name; p. 6, paragraph 0075], and a third field for storing a type associated with the sub-component [a "javaType" attribute; p. 6, paragraph 0075].

16. As to claim 13, Russell teaches negotiating the list by accepting a plurality of object types received from a first process [added to a mapping registry], the accepted object types becoming known object types identified within the list [programmatically-generated mappings are automatically added to a mapping registry; p. 2, paragraph 0037].

17. As to claim 14, Russell teaches negotiating the list by receiving an identifier for a file and having the list include object types identified within the file [encodingStyle" attribute where that default value identifies the SOAP encoding schema; p. 6, paragraph 0074].

18. As to claim 16, Russell teaches limiting the hierarchy of sub-components [Each map element has six attributes; p. 6, paragraph 0074] by defining a property set that identifies individual properties of the object [attributes; p. 6, paragraph 0080], wherein decomposing the object comprises decomposing the identified individual properties of the object [For the object to be serialized, the values of a "qname" attribute and of a "javaType" attribute are programmatically generated; p. 6, paragraph 0075].

19. As to claim 17, Russell teaches limiting the hierarchy of sub-components by identifying a specified property within the object [a type attribute 932 is set to match the name of the class being serialized; p. 6, paragraph 0080], wherein decomposing the object comprises decomposing the specified property [element is created for the endDate field, as shown at 970, where this element has a type attribute 971; p. 7, paragraph 0085].

20. As to claim 27, this is a system claims that corresponds to product claim 5; note the rejection to claim 5 above, which also meets the limitations of this system claim.

21. As to claim 30, Russell teaches just the known objects are serialized in the serializing act [p. 4, paragraph 0057].

22. As to claim 31, Russell teaches the method further comprises negotiating with the remote entity to determine which object types are known by the remote entity [p. 2, paragraph 0037].

23. As to claim 32, Russell teaches negotiating with the remote entity to determine which object types are known by the remote entity [p. 2, paragraph 0037].

Claim Rejections - 35 USC § 103

24. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

25. This application currently names joint inventors. In considering patentability of the claims under 35 U.S.C. 103(a), the examiner presumes that the subject matter of the various claims was commonly owned at the time any inventions covered therein were made absent any evidence to the contrary. Applicant is advised of the obligation under 37 CFR 1.56 to point out the inventor and invention dates of each claim that was not commonly owned at the time a later invention was made in order for the examiner to consider the applicability of 35 U.S.C. 103(c) and potential 35 U.S.C. 102(e), (f) or (g) prior art under 35 U.S.C. 103(a).

26. **Claims 9, 10 and 12 are rejected under 35 U.S.C. 103(a) as being unpatentable over Russell in view of U.S. Patent Application Publication No. 2003/0182308 to Ernst et al. [hereinafter Ernst, previously cited].**

27. As to claim 9, Russell teaches property bag [p. 6, paragraph 0074] but does not teach the property bag comprising a hash table.

However, Ernst teaches transforming data from one format to another [p. 5, paragraph 0056] based on data schema [p. 5, paragraph 0057], a property bag [a set of properties of that object; p. 4, paragraph 0039] comprising a hash table [lookup table; p. 4, paragraph 0039].

It would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine the teachings of Ernst and Russell because Ernst's teachings allows content schema evolution while maintaining operation based on already stored content data [p. paragraph 0011 of Ernst] and simplifies maintenance and takes some burden of the setup phase [p. 10, paragraph 0118 of Ernst].

28. As to claim 10, Russell does not teach a key for each entry in the hash table comprises a name for the sub-component associated with the entry.

However, Ernst teaches a key [ID] for each entry in the hash table [selecting from the property tables all rows with the object ID associated to the content object; p. 11, paragraph 0132] comprises a name for the sub-component associated with the entry [properties are internally represented by identifiers; p. 11, paragraph 0128].

It would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine the teachings of Ernst and Russell because Ernst's teachings provide unique invariable identifiers for the content types and/or properties [p. 5, paragraph 0051 of Ernst].

29. As to claim 12, Russell does not teach version numbers for an object type list.

However, Ernst teaches negotiating [pp. 4 – 5, paragraph 0047] the known object types identified within list by receiving a version number of a first list [label referring to a content schema version; p. 4, paragraph 0043] available to a first process [a label is used to record a set of versions that fulfill some interobject integrity constraints; p. 4,

paragraph 0043], comparing the version number to another version number of a second list available to the remote entity [a version is directly linked from many sources, a modified version would only be linked after changing all referring resources; p. 4, paragraph 0044], and determining the list based on the comparison [generating new version of these resources; p. 4, paragraph 0044].

It would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine the teachings of Ernst and Russell because Ernst's teachings provide a unified approach to handle content schema evolution and content generation [p. 6, paragraph 0061 of Ernst].

30. **Claims 15 and 18 – 25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Russell in view of U.S. Patent Application Publication No. 2005/0154978 to Albornoz et al. [hereinafter Albornoz, previously cited].**

31. As to claim 15, Russell does not teach limiting the hierarchy of sub-components by specifying a pre-determined depth for the hierarchy.

However, Albornoz teaches Serialization of the objects within an XML data structure API class [p. 3, paragraph 0031], hierarchical data structures with one or more layers of sub-elements [p. 4, paragraph 0043], limiting the hierarchy of sub-components by specifying a pre-determined depth for the hierarchy [a minimum number of elements in the array is not met and/or the maximum number of elements in the array is exceeded; p. 7, paragraph 0062], wherein decomposing the object comprises

decomposing the object to the pre-determined depth [If the maxOccurs for the current sub-element is determined to not be greater than one; p. 7, paragraph 0067].

It would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine the teachings of Albornoz and Russell because Albornoz's teaching raises an error if a maximum number of elements is exceeded or minimum number of elements is not met [p. 7, paragraph 0062 of Albornoz]. The raised error would stop processing when the maximum number of elements is exceeded and this would prevent out of memory errors.

32. As to claim 18, Russell as modified by Albornoz teaches limiting the hierarchy of sub-components by specifying a pre-determined number [a minimum number of elements in the array is not met and/or the maximum number of elements in the array is exceeded; p. 7, paragraph 0062 of Albornoz] that limits the known objects that are serialized into the serialized package by the number [If the maxOccurs for the current sub-element is determined to not be greater than one; p. 7, paragraph 0067 of Albornoz]. As to the motivation for combining Russell and Albornoz, see the rejection to claim 15 above.

33. As to claim 19, Russell teaches the invention substantially as claimed including at least one storage computer-readable medium [p. 8, paragraph 0096] having computer executable instructions that provide a method for receiving a package

representing a computer-readable object transmitted across a remote boundary [p. 7, paragraph 0088], the method comprising:

receiving a serialized package [serialize the objects to be passed as input to the Web service so that these objects and their data can be passed to the Web service; p. 3, paragraph 0048] from a remote entity [client code, p. 3, paragraph 0048];

identifying a hierarchy of sub-components ["mappings" element 800 contains a child "map" element for each mapping being generated for this JavaBean; p. 6, paragraph 0074], the hierarchy representing an object of a first type [type mappings; p. paragraph 0077];

for each sub-component: identifying a type associated with the sub-component [mappings generated according to the present invention may be used when deserializing messages (i.e., when transforming flattened, or non-hierarchical, data structures back into their object format); p. 2, paragraph 0038];
determining whether that identified type is within a list of know object types [deserialization leverages the mappings that were previously created and stored in the mappings registry, as needed; p. 7, paragraph 0092];

responding to the determining, wherein the responding comprises instantiating an object of the type [deserialization process uses the stored type mappings to generate an object version of the message; p. 2, paragraph 0021], wherein the instantiating and populating are performed when the identified type is within the list of known object types [deserialization leverages the mappings that were previously created and stored in the mappings registry; p. 7, paragraph 0092]. Although Russell teaches the invention

substantially, Russell does not specifically teach populating at least one property of the object with information obtained from within the serialized package.

However, Albornoz teaches serialization of the objects within an XML data structure API class [p. 3, paragraph 0031], hierarchical data structures with one or more layers of sub-elements [p. 4, paragraph 0043], instantiating an object of a type and populating at least one property of the object with information obtained from within the serialized package [creates, at step 1004, an object that corresponds to the received schema; p. 7, paragraph 0065].

It would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine the teachings of Albornoz and Russell because Albornoz's teaching provides easy manipulation within dynamic languages of data structures that reflect the structure of content models defined by XML schemas [pp. 5 – 6, paragraph 0053 of Albornoz].

34. As to claim 20, Russell teaches the list includes the first type as one of the known object types [deserialization leverages the mappings that were previously created and stored in the mappings registry; p. 7, paragraph 0092].

35. As to claim 21, Russell teaches the at least one sub-component comprises an unknown object having a type unidentified within the list [a "missing" type mapping; p. 2, paragraph 0036 and p. 4, paragraph 0059].

36. As to claim 22, Russell teaches a first process [Web service 1030; p. 7, paragraph 0088] on a system receives the serialized package [p. 4, paragraph 0057] and the remote entity comprises another process [object-based client code; p. 3, paragraph 0048] on the system [client/server; p. 3, paragraph 0043].

37. As to claim 23, Russell teaches a first process [Web service 1030; p. 7, paragraph 0088] on a system receives the serialized package [p. 4, paragraph 0057] and the remote entity comprises another process [object-based client code; p. 3, paragraph 0048] on another system [client/server; p. 3, paragraph 0043].

38. As to claim 24, Russell teaches a first application domain [client/server networks; p. 3, paragraph 0043] executing within a process [Web service 1030; p. 7, paragraph 0088] receives the serialized package [p. 4, paragraph 0057] and the remote entity [object-based client code; p. 3, paragraph 0048] comprises another application domain [client/server networks; p. 3, paragraph 0043] within the process.

39. As to claim 25, Russell teaches the serialized package comprises an XML document [SOAP XML format; p. 4, paragraph 0057].

Conclusion

40. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

CONTACT INFORMATION

41. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Li B. Zhen whose telephone number is (571) 272-3768. The examiner can normally be reached on Mon - Fri, 8:30am - 5pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, William Thomson can be reached on 571-272-3718. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

Li B. Zhen
Primary Examiner
Art Unit 2194

lbz

/Li B. Zhen/
Primary Examiner, Art Unit 2194